

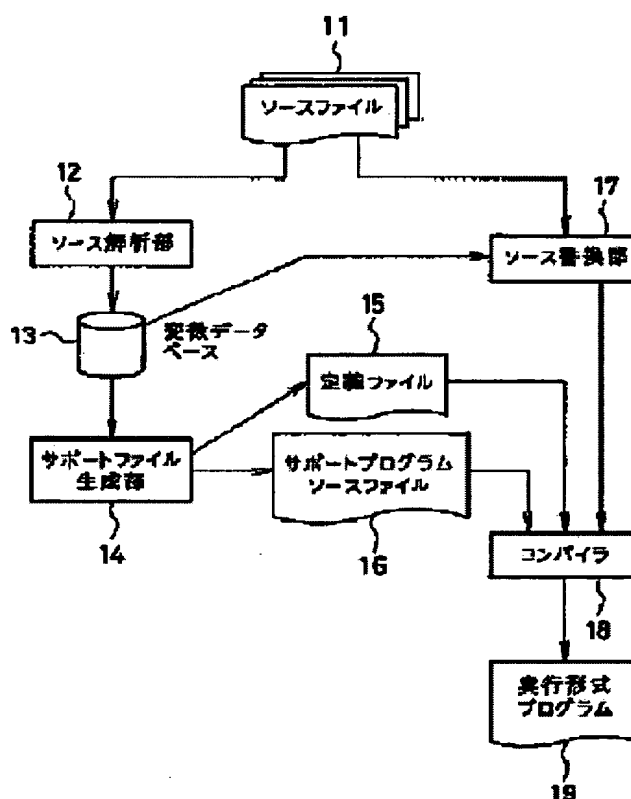
AUTOMATIC RE-ENTRANT METHOD FOR NON-RE-ENTRANT PROGRAM AND SYSTEM THEREFOR

Patent number: JP11272475
 Publication date: 1999-10-08
 Inventor: MURAKAMI TAKUYA
 Applicant: NIPPON ELECTRIC CO
 Classification:
 - international: G06F9/45; G06F9/46
 - european:
 Application number: JP19980092447 19980320
 Priority number(s): JP19980092447 19980320

Report a data error here

Abstract of JP11272475

PROBLEM TO BE SOLVED: To provide a method for attaining automatic conversion from a non-re-entrant program into a re-entrant program. **SOLUTION:** A source file 11 is analyzed by a source analyzing part 12, and the declaring part of a global variable and a static variable is extracted, and registered in a variable data base 13. A support file generating part 14 generates a definition file 15 in which a definition sentence necessary for a source file after rewriting is described based on the variable data base and a support program source file 16 in which a program such as an initialization function necessary for the initialization of the variable is described. A source rewriting part 17 deletes the definition of the global variable and the static variable registered in the variable data base from the source file, rewrites the variable using part and changes this so that a substitute variable held for each rewiring task can be used, and generates a re-entranced source file. Then, the compile link of this source file with the definition file and the support program is operated, and a re-entrant execution format program is generated.



Data supplied from the esp@cenet database - Worldwide

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平11-272475

(43) 公開日 平成11年(1999)10月8日

(51) Int. CL ⁸	識別記号	P 1	
G 0 6 F 9/45		G 0 6 F 9/44	3 2 2 A
9/46	3 4 0	9/46	3 4 0 F

審査請求 有 請求項の数 4 F D (全 10 頁)

(21) 出願番号 特願平10-92447

(22) 出願日 平成10年(1998)3月20日

(71) 出願人 000004237

日本電気株式会社

東京都港区芝五丁目7番1号

(72) 発明者 村上 卓弥

東京都港区芝五丁目7番1号 日本電気株式会社社内

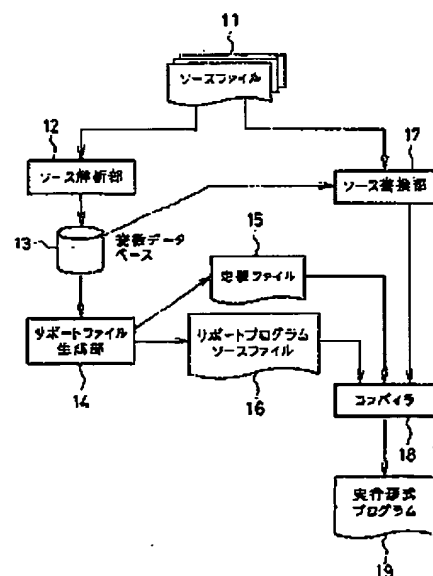
(74) 代理人 弁理士 加藤 朝道

(54) 【発明の名称】 非リエントラントプログラムの自動リエントラント化方法及びシステム

(57) 【要約】

【課題】 非リエントラントプログラムからリエントラントプログラムへの自動変換を行う方法の提供。

【解決手段】 ソースファイルをソース解析部で解析し、グローバル変数及びスタティック変数の宣言部を抽出して変数データベースに登録し、サポートファイル生成部で、変数データベースを基に書き換え後のソースファイルに必要な定義文を記述した定義ファイルと、変数の初期化に必要な初期化関数などのプログラムを記述したサポートプログラムソースファイルを生成し、ソース書換部でソースファイルから変数データベースに登録されたグローバル変数及びスタティック変数の定義の削除を行い、変数の使用箇所を書き換えタスク毎に保持される代替変数を使用するように変更を行い、リエントラント化されたソースファイルを生成し、これを定義ファイル、サポートプログラムをコンパイルリンクしてリエントラント型の実行形式プログラムを生成する。



(2)

特開平11-272475

1

2

【特許請求の範囲】

【請求項1】(a)非リエントラントプログラムのソースファイルを解析し、グローバル変数、及びスタティック変数の宣言部を抽出してこれを変数データベースに登録し、

(b)前記変数データベースの情報を基にして、書き換え後のソースファイルで必要とされる定義文などを記述した定義ファイルと、変数の初期化に必要な初期化関数及び代替変数アクセス関数などのプログラムを記述したサポートプログラムソースファイルを生成し、

(c)前記ソースファイルから、前記変数データベースに登録されたグローバル変数、及びスタティック変数の定義の削除を行い、変数の使用箇所を書き換え、タスク毎に保持される代替変数を使用するように変更を行い、リエントラント化されたソースファイルを生成し、

(d)前記リエントラント化されたソースファイルと、前記定義ファイル、及び、前記サポートプログラムをコンパイル及びリンクしてリエントラント型の実行形式プログラムを生成する、ことを特徴とする非リエントラントプログラムの自動リエントラント化方法。

【請求項2】非リエントラントプログラムのソースファイルを解析し、グローバル変数、及びスタティック変数の宣言部を抽出し、これを変数データベースに登録するソース解析手段と、

前記変数データベースの情報を基にして、書き換え後のソースファイルで必要とされる定義文などを記述した定義ファイルと、変数の初期化に必要な初期化関数及び代替変数アクセス関数などのプログラムを記述したサポートプログラムソースファイルとを生成するサポートファ

イル生成手段と、
前記ソースファイルから、前記変数データベースに登録されたグローバル変数、及びスタティック変数の定義の削除を行い、変数の使用箇所を書き換え、タスク毎に保持される代替変数を使用するように変更を行うソース書換手段と、

を備え、

前記ソース書換手段で書換えられたソースファイルと、前記定義ファイル、及び前記サポートプログラムソースファイルをコンパイル及びリンクしてリエントラント型の実行形式プログラムを生成する、ことを特徴とする非リエントラントプログラムの自動リエントラント化システム。

【請求項3】前記ソース書換手段が、前記ソースファイルを走査し、変数が見つかった場合に、該変数が前記変数データベースに登録されているか否かを検索し、登録されている場合には、これが変数の宣言部であるか調べ、変数の宣言部である場合には前記ソースファイルから該宣言部を削除し、一方宣言部でなければ、前記ソースファイルにおける前記変数へのアクセスを、前記サポ

ートプログラムソースファイルに定義されている代替変数アクセス関数へ置き換える、ことを特徴とする請求項2記載の非リエントラントプログラムの自動リエントラント化システム。

【請求項4】(a)非リエントラントプログラムのソースファイルを解析し、グローバル変数、及びスタティック変数の宣言部を抽出し、これを変数データベースに登録する処理、

(b)前記変数データベースの情報を基にして、書き換え後のソースファイルで必要とされる定義文などを記述した定義ファイルと、変数の初期化に必要な初期化関数及び代替変数アクセス関数などのプログラムを記述したサポートプログラムソースファイルを生成する処理、

(c)前記ソースファイルから、前記変数データベースに登録されたグローバル変数、及びスタティック変数の定義の削除を行い、変数の使用箇所を書き換え、タスク毎に保持される代替変数を使用するように変更を行う処理、

の上記(a)～(c)の各処理をコンピュータで実行させるためのプログラムを記録した記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ソフトウェア開発支援システムに関し、特に非リエントラント構造のプログラムをリエントラント構造のプログラムに自動変換する方法及び方式に関する。

【0002】

【従来の技術】情報処理システムでは、メモリ空間が全タスクで共有される方式をとっているものがある。このようなシステムで、複数のタスクから呼び出される(callee)ようなリエントラントなプログラムを作成する場合、以下に説明する2つの方法が知られている。

【0003】その第1の方法は、このプログラムのコード領域とグローバル変数及びスタティック変数領域を、このプログラムを呼び出すプログラム毎に用意するというものである。しかし、この方法は、メモリの使用効率が悪くなるため、メモリの制約が厳しいシステムでは採用できない場合がある。

【0004】これを解決するための方法(第2の方法)として、コード領域をこのプログラムを呼び出す全プログラム間で共有させればよい。この場合、このプログラムのコード領域は1つだけあればよいのでメモリの使用効率がよい。

【0005】しかしながら、コード領域を共有させると、メモリ空間が共有されているシステムでは、グローバル変数及びスタティック変数領域まで共有されてしまうため、プログラムが非リエントラントになってしまう。複数のタスクから呼び出して使用することができない。

(3)

特開平11-272475

3

4

【0006】これを回避するためには、プログラム内で使用するグローバル変数、及びスタティック変数の代替となる変数をタスク毎に用意し、タスク毎に自動的に切り替えて使用する機構を導入する必要がある。そして、これを行うには、プログラムのソースファイル自体を修正する必要がある。

【0007】具体的には、プログラムのソースファイルにおいて、グローバル変数とスタティック変数の宣言部の削除と、これらの変数の使用箇所の書き換えが必要である。

【0008】

【発明が解決しようとする課題】従来、この書き換えは、全て手作業で行う必要があったが、手作業で行うと、以下に示すような問題が生じることになる。

【0009】(1) プログラムが巨大な場合、手作業では莫大な労力、作業工数を要する。

【0010】(2) 手作業のためミスが発生しやすい。

【0011】(3) ミスを犯した場合、見つけにくいバグの原因となる。

【0012】(4) 修正前と修正後の2つのプログラムを保守管理する必要があるため、保守性が低下する。

【0013】このため、この作業を自動化したいという要求があった。この書き換え作業を自動化することができれば、以上の問題は、全て解決することになる。

【0014】したがって、本発明は、上記技術的認識に基づき創案されたものであって、その目的は、非リエントラント型のプログラムからリエントラント型のプログラムへの自動変換を行う方法及びシステムを提供することにある。

【0015】

【課題を解決するための手段】前記目的を達成するため、本発明は、(a) 非リエントラントプログラムのソースファイルを解析し、グローバル変数、及びスタティック変数の宣言部を抽出してこれを変数データベースに登録し、(b) 前記変数データベースの情報を基にして、書き換え後のソースファイルで必要とされる定義文などを記述した定義ファイルと、変数の初期化に必要な初期化関数及び代替変数アクセス関数などのプログラムを記述したサポートプログラムソースファイルを生成し、(c) 前記ソースファイルから、前記変数データベースに登録されたグローバル変数、及びスタティック変数の定義の削除を行い、変数の使用箇所を書き換え、タスク毎に保持される代替変数を使用するように変更を行い、リエントラント化されたソースファイルを生成し、(d) リエントラント化されたソースファイルと、前記定義ファイル、及びサポートプログラムファイルを参照してコンパイル及びリンクしリエントラント型の実行形式プログラムを生成するようにしたものである。

【0016】

【発明の実施の形態】本発明は、非リエントラント型の

プログラムを、メモリ空間が共有されているシステム用のリエントラント型のプログラムに修正するために、自動的にリエントラント化するための手段を提供するものである。より詳細には、本発明は、プログラム内で使用される、リエントラント時に問題になるグローバル変数、及びスタティック変数の削除を、ソースプログラムレベルで行い、このように変更されたソースプログラムに対して、もとのプログラムと同等の動作を行うようにさせるために必要なコード(ソースコード)の追加を自動的に行うものである。

【0017】また、本発明は、プログラムをROM(読み出し専用メモリ)対応プログラムに自動的に修正するための方法も提供する。

【0018】本発明の実施の形態について図面を参照して説明する。図1は、本発明の実施の形態の処理動作を説明するための図である。図1を参照すると、ソースファイル11はリエントラント化の対象となる非リエントラントプログラムのソースコードであり、本発明を適用することにより、このソースファイル11からリエントラント化がなされた、実行形式プログラム19が生成される。本発明は、その好ましい実施の形態において、ソース解析部12、サポートファイル生成部14、ソース解析部17を備える。これらの各部の処理は、コンパイラ18等と同様コンピュータ上で実行されるプログラムで実現される。

【0019】まず、全てのソースファイル11をソース解析部12を用いて解析し、グローバル変数、及びスタティック変数の宣言部を抽出し、これを変数データベース13に登録する。

【0020】次に、この変数データベース13を基にして、サポートファイル生成部14で、定義ファイル15、及びサポートプログラムソースファイル16を生成する。定義ファイル15には、書き換え後のソースファイルで必要とされる定義文などが記述されている。また、サポートプログラムソースファイル16には、変数の初期化に必要な処理などのプログラムが納められる。

【0021】その後、変数データベース13の情報を基にして、ソースファイル11をソース書換部17で書き換える。この際、変数データベース13に登録されたグローバル変数、及びスタティック変数の定義の削除を行い、変数の使用箇所を書き換えて、タスク毎に保持される代替変数を使用するように変更を行う。

【0022】この時点でリエントラント化されたソースファイルが全て生成されている。このソースファイルを、定義ファイル、サポートプログラムソースファイルとともにコンパイラ18でコンパイルし、リンクでリンクすることでリエントラント化された実行形式プログラム19が得られる。

【0023】

【実施例】上記した本発明の実施の形態についてさらに

(3)

特開平11-272475

3

4

【0006】これを回避するためには、プログラム内で使用するグローバル変数、及びスタティック変数の代替となる変数をタスク毎に用意し、タスク毎に自動的に切り替えて使用する機構を導入する必要がある。そして、これを行うには、プログラムのソースファイル自体を修正する必要がある。

【0007】具体的には、プログラムのソースファイルにおいて、グローバル変数とスタティック変数の宣言部の削除と、これらの変数の使用箇所の書き換えが必要である。

【0008】

【発明が解決しようとする課題】従来、この書き換えは、全て手作業で行う必要があったが、手作業で行うと、以下に示すような問題が生じることになる。

【0009】(1) プログラムが巨大な場合、手作業では莫大な労力、作業工数を要する。

【0010】(2) 手作業のためミスが発生しやすい。

【0011】(3) ミスを犯した場合、見つけにくいバグの原因となる。

【0012】(4) 修正前と修正後の2つのプログラムを保守管理する必要があるため、保守性が低下する。

【0013】このため、この作業を自動化したいという要求があった。この書き換え作業を自動化することができれば、以上の問題は、全て解決することになる。

【0014】したがって、本発明は、上記技術的認識に基づき創案されたものであって、その目的は、非リエントラント型のプログラムからリエントラント型のプログラムへの自動変換を行う方法及びシステムを提供することにある。

【0015】

【課題を解決するための手段】前記目的を達成するため、本発明は、(a) 非リエントラントプログラムのソースファイルを解析し、グローバル変数、及びスタティック変数の宣言部を抽出してこれを変数データベースに登録し、(b) 前記変数データベースの情報を基にして、書き換え後のソースファイルで必要とされる定義文などを記述した定義ファイルと、変数の初期化に必要な初期化関数及び代替変数アクセス関数などのプログラムを記述したサポートプログラムソースファイルを生成し、(c) 前記ソースファイルから、前記変数データベースに登録されたグローバル変数、及びスタティック変数の定義の削除を行い、変数の使用箇所を書き換え、タスク毎に保持される代替変数を使用するように変更を行い、リエントラント化されたソースファイルを生成し、(d) リエントラント化されたソースファイルと、前記定義ファイル、及びサポートプログラムファイルを参照してコンパイル及びリンクしリエントラント型の実行形式プログラムを生成するようにしたものである。

【0016】

【発明の実施の形態】本発明は、非リエントラント型の

プログラムを、メモリ空間が共有されているシステム用のリエントラント型のプログラムに修正するために、自動的にリエントラント化するための手段を提供するものである。より詳細には、本発明は、プログラム内で使用される、リエントラント時に問題になるグローバル変数、及びスタティック変数の削除を、ソースプログラムレベルで行い、このように変更されたソースプログラムに対して、もとのプログラムと同等の動作を行うようにさせるために必要なコード(ソースコード)の追加を自動的に行うものである。

【0017】また、本発明は、プログラムをROM(読み出し専用メモリ)対応プログラムに自動的に修正するための方法も提供する。

【0018】本発明の実施の形態について図面を参照して説明する。図1は、本発明の実施の形態の処理動作を説明するための図である。図1を参照すると、ソースファイル11はリエントラント化の対象となる非リエントラントプログラムのソースコードであり、本発明を適用することにより、このソースファイル11からリエントラント化がなされた、実行形式プログラム19が生成される。本発明は、その好ましい実施の形態において、ソース解析部12、サポートファイル生成部14、ソース解析部17を備える。これらの各部の処理は、コンパイラ18等と同様コンピュータ上で実行されるプログラムで実現される。

【0019】まず、全てのソースファイル11をソース解析部12を用いて解析し、グローバル変数、及びスタティック変数の宣言部を抽出し、これを変数データベース13に登録する。

【0020】次に、この変数データベース13を基にして、サポートファイル生成部14で、定義ファイル15、及びサポートプログラムソースファイル16を生成する。定義ファイル15には、書き換え後のソースファイルで必要とされる定義文などが記述されている。また、サポートプログラムソースファイル16には、変数の初期化に必要な処理などのプログラムが納められる。

【0021】その後、変数データベース13の情報を基にして、ソースファイル11をソース変換部17で書き換える。この際、変数データベース13に登録されたグローバル変数、及びスタティック変数の定義の削除を行い、変数の使用箇所を書き換えて、タスク毎に保持される代替変数を使用するように変更を行う。

【0022】この時点でリエントラント化されたソースファイルが全て生成されている。このソースファイルを、定義ファイル、サポートプログラムソースファイルとともにコンパイラ18でコンパイルし、リンクでリンクすることでリエントラント化された実行形式プログラム19が得られる。

【0023】

【実施例】上記した本発明の実施の形態についてさらに

(4)

特開平11-272475

5

6

詳細に説明すべく、本発明の実施例について以下に説明する。以下では、C言語で記述されたソースプログラムを対象とする。

【0024】図2は、本発明の一実施例の構成を示すブロック図である。図2を参照すると、本実施例は、ソースファイル記憶領域21と、プリプロセッサ22と、ソース解析部23と、変数データベース24と、サポートファイル生成部25と、ソース書換部26と、コンパイラ27と、リンカ28と、実行形式プログラムファイル記憶領域29と、を備えて構成されている。

【0025】ソースファイル記憶領域21は、プログラムのソースプログラムファイルを記憶しておく領域である。

【0026】プリプロセッサ22は、Cプログラム用のプリプロセッサ（前処理系）である。

【0027】ソース解析部23は、ソースプログラム中のグローバル変数、及びスタティック変数の定義部を抽出し、この情報を変数データベース24に登録する。

【0028】サポートファイル生成部25は、変数データベース24に格納された情報から、ソース書換部26およびコンパイラ27で必要になるファイルを生成する。

【0029】ソース書換部26は、ソースプログラム中のグローバル変数、及びスタティック変数の宣言部および使用部（プログラム中でグローバル変数、スタティック変数をアクセスする部分）を置き換えたものを出力する。

【0030】コンパイラ27は、この出力をコンパイルしてオブジェクトコードを出力し、リンカ28はコンパイラ27が生成したオブジェクトコードをリンクして、実行形式プログラムを生成する。リンカ28が生成する実行形式プログラムファイルは、実行形式プログラムファイル記憶領域29に保存される。

【0031】図3は、本発明の一実施例の処理を説明するための図である。図2及び図3を参照して、本発明の一実施例の動作について以下に説明する。図3において、変数データベース34は図2の変数データベース24に対応し、プリプロセッサ22、ソース解析部23、ソース書換部26等は、図2のプリプロセッサ22、ソース解析部23、ソース書換部26等の処理を表している。

【0032】まず、ソースファイル31を、プリプロセッサ22でプリプロセス（前処理）する（32）。その後、ソース解析部23で解析処理を行い（33）、スタティック変数データベース34を出力する。

【0033】図4は、本発明の一実施例におけるソース解析部23の処理フローを説明するための流れ図である。図4の流れ図を参照して、ソース解析部23の処理について説明する。

【0034】初めに、ステップA1で、ソースファイル31を構文解析し、変数定義を探す。

【0035】次にステップA2で、ファイルの終端（E

OF）に達したかどうか否かを判定し、終端に達したならば処理を終了する。

【0036】変数が見つかったら、ステップA3で、その変数が関数内で定義されたローカル（local）なものかどうかを調べ、ローカルだった場合、ステップA4で、その変数に、さらにstatic（スタティック）キーワードがついているかどうかを調べ、付いていなければ、その変数は無視してステップA1に戻る。

【0037】ステップA3で変数がローカルでない場合、及びローカルである場合で且つスタティックと定義されている場合にはステップA5に進む。

【0038】次に、ステップA5でconst（コンスタント）キーワードが付いているかどうかを調べる。constが付いている場合、その変数は定数であって書き換えは不可能であるため、リエントラント化の考慮を行う必要はない。よって、この変数は無視してステップA1に戻る。

【0039】次に、ステップA5において変数がconstでない場合、ステップA6で、変数の初期化処理があるかどうかを調べ、初期化処理があれば、ステップA7で、その初期化処理を取り出す。

【0040】最後に、ステップA8で変数データベース34に登録し、ステップA1に戻る。

【0041】一例として、図6に示したソースプログラムを解析した結果得られる変数データベースの例を図7に示す。

【0042】データベースには、変数の型名（整数型int、文字型char等）、スコープ（変数がプログラム中で有効となる範囲を表す情報、global、sub()等）、変数名、初期化処理を記録する。

【0043】再び図2及び図3を参照すると、また、この変数データベース24から、サポートファイル生成部25で必要なヘッダファイル35とサポートプログラムソースファイル36を生成する。

【0044】ヘッダファイル35には、グローバル変数、及びスタティック変数の代替変数をひとまとめにした構造体の定義、およびマクロ、外部関数定義が入る。

【0045】また、サポートプログラムソースファイル36には、初期化が必要なグローバル変数及びスタティック変数の初期化処理ルーチンと、代替変数アクセスルーチン処理が入る。

【0046】図7のデータベースから作成されるヘッダファイル（qsdef.h）の例を図8に、サポートプログラムファイル（qsint.c）の例を図9に示す。

【0047】なお、変数名が衝突しないように名前付けを行う必要がある。この例では、例えば、図6に示したソースプログラムの関数sub中の変数cは、図8に示すように、_sub_cという代替の名前を付けている。なお、この例では、ヘッダファイル（qsdef.h）には、グローバル変数（_global_a、_global_b）、及びスタティック

50

(5)

特開平11-272475

7

変数（_sub_d）の代替変数をひとまとめにした構造体（struct GSDat）の定義、およびマクロ（#define GSDAT（x）（qsd（x）->（x）））、外部関数定義（extern struct GSDat *qsd（void））が導入される。

【0048】またサポートプログラムファイル（qsimr.c）では、ヘッダファイル（qsd.h）をインクルードファイルとし（#include）、初期化処理関数（InitGSDat）として、ポインタ変数pで参照される代替変数構造体のうち（p->global_b）を20、d（p->_sub_d）を5に設定している。そして、代替変数構造体アクセス関数（GetGSDat）は、グローバル変数及びスタティック変数のタスク毎に固有な代替変数の構造体を返す関数であり、タスク固有のID（識別）番号から対応する代替変数の構造体GSDatへのポインタpを関数値として返却する。すなわち、この構造体GSDatへのポインタpにはタスクID値からタスク毎のGSDatのテーブルが探索され、もしpがnullの場合、pを新たに割り当ててテーブルに登録し、初期化関数InitGSDat（引数p）を呼び出して初期化処理を行った後に返却する（return p）。

【0049】再び図2及び図3を参照すると、次に、プリプロセッサ済みのソースファイルをソース変換部26を使って書き換える（37）。

【0050】図5は、本発明におけるソース変換部26の処理フローを説明するための流れ図である。図5を参照して、ソース変換部26の処理について説明する。

【0051】まず、ステップB1、B2でソースコードの先頭から順に変数定義を全て探す。変数が見つかったら、ステップB3、B4で、それが変数データベース24に登録されているか否かを調べる。登録されていた場合、ステップB5で、それが変数の宣言部であるか否かを調べ、宣言部であれば、ステップB6で削除する。

【0052】宣言部でなければ、ステップB7で書き換えを行い、代替変数へのアクセスに切り替える。

【0053】これは、サポートプログラム中に含まれるアクセスルーチンの呼び出しに置き換えることで行う。

【0054】このようにして、図6に示したソースプログラムを書き換えた例を図10に示す。

【0055】図9のサポートプログラムファイル中には、関数「GetGSDat」が含まれている。この関数は、グローバル変数及びスタティック変数のタスク毎に固有な代替変数のセット（構造体）を返す関数であり、タスク固有のID番号から対応する代替変数のセットを返すようになっている。

【0056】ソース変換後のプログラムでは、全てこの関数を経由して代替変数にアクセスするようになっている。

【0057】このため、このプログラムを使用するタスク毎に代替変数が切り替わることになり、プログラムがリエントラント化される。

8

【0058】再び図2及び図3を参照すると、この変換後のソースファイルとヘッダファイル35を用いて再びプリプロセッサを行い（38）、コンパイラ27を用いてコンパイルする（39）。

【0059】また、サポートプログラムソースファイル36も同様にコンパイルする。得られたオブジェクトファイルをリンカ28を使ってリンクし（310）、最終的にリエントラントな実行形式プログラムが得られる（311）。

10 【0060】以上、C言語の場合の例に説明したが、ソース解析部、サポートファイル生成部、ソース変換部を、他言語対応のものに置き換えることにより、所望の言語への適用が可能である。

【0061】本発明の第二の実施例として、C++言語に適用する場合について説明する。

【0062】C++言語に適用する場合、以下の2つの方法がある。

1. C++ソースをCに変換し、C言語の場合と同様に処理する。
2. C++ソースをそのまま処理する。

【0063】第1の方法については、AT&T社製によるcfrontというC++トランスレータを使うことにより、C++をCのソースファイルに変換することができる。C言語のソースファイルに対して、上記した本発明の実施例がそのまま適用される。

【0064】第2の方法については、ほぼCの場合と同じように処理できるが、C++ソースのclass（クラス）内で宣言されているスタティック変数が使用されることがあるので、それを考慮する必要がある。そのため、図2のソース解析部3のアルゴリズムを修正し、class内スタティック変数も変数データベースに登録するようにする。

【0065】また、C++ではグローバル及びスタティックなオブジェクトが存在した場合、サポートプログラム内の初期化処理において、このオブジェクトのコンストラクタを呼び出す処理を追加する。

【0066】さらに、本発明は、プログラムをROM対応プログラムに修正する際にも応用することができる。プログラムをROM化する場合、初期化に必要なグローバル変数及びスタティック変数をROM化することができないため、これを取り除かねばならない。

【0067】この場合、上記した本発明の実施例の方法をそのまま適用すれば、これらの変数を取り除くことができるため、簡単にROM対応を行うことができる。

【0068】本発明をC言語に適用する場合において、図2のソース解析部23を一部修正し、変数データベース24には、初期化が必要な変数のみを登録するようにしてもよい。

【0069】

50 【発明の効果】以上説明したように、本発明によれば、

(5)

特開平11-272475

9

15

下記記載の効果を奏する。

【0070】本発明の第1の効果は、リエントラント化のための入手による変換作業・労力を完全に不要とすることができ、開発効率を向上し作業工数を特段に削減することができる、ということである。

【0071】その理由は、本発明においては、リエントラントプログラムへの変換を完全に自動化しており、入手作業を一切必要としないためである。

【0072】本発明の第2の効果は、プログラムのリエントラント化に伴うバグの発生が生じない、ということである。

【0073】その理由は、本発明においては、変換にエディット（編集）作業等の、手作業が介入せず、変数名の見落とし、書き換え間違いなどの人為ミスが混入することがないためである。

【0074】本発明の第3の効果は、保守性を向上する、ということである。

【0075】その理由は、本発明においては、ソースファイルを自動修正できるため、修正後のソースファイルを保守管理する必要がないためである。

【0076】本発明の第4の効果は、開発環境および実行環境への依存度が低く、汎用性、可用性が高い、ということである。

【0077】その理由は、本発明においては、ソースファイルに対して修正を行うのみであるため、特定のコンパイラやOS（オペレーティングシステム）に依存しないためである。

【図面の簡単な説明】

【図1】本発明の実施の形態を説明するための図である。

【図2】本発明の一実施例の構成を示すブロック図である。

【図3】本発明の一実施例の処理の流れを示す図である。

【図4】本発明の一実施例におけるソース解析部の処理フローを示す流れ図である。

【図5】本発明の一実施例におけるソース書換部の処理フローを示す流れ図である。

【図6】本発明の一実施例を説明するための図であり、非リエントラントソースプログラムの一例を示す図である。

【図7】本発明の一実施例を説明するための図であり、変数データベースの一例を示す図である。

【図8】本発明の一実施例を説明するための図であり、サポートファイル生成部で生成されたヘッダファイルの一例を示す図である。

【図9】本発明の一実施例を説明するための図であり、サポートファイル生成部で生成されたサポートプログラムファイルの一例を示す図である。

【図10】本発明の一実施例を説明するための図であり、リエントラント化されたソースプログラムの一例を示す図である。

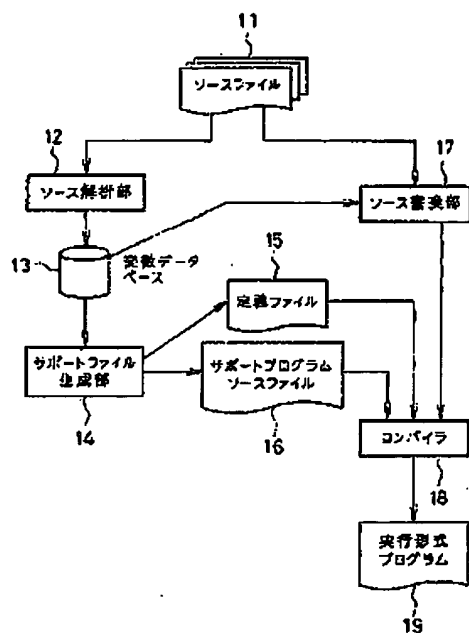
【符号の説明】

- 11 ソースファイル
- 12 ソース解析部
- 13 変数データベース
- 14 サポートファイル生成部
- 15 定義ファイル
- 16 サポートプログラムソースファイル
- 17 ソース書換部
- 18 コンパイラ
- 19 実行形式プログラム
- 21 ソースファイル記憶領域
- 22 プリプロセッサ
- 23 ソース解析部
- 24 変数データベース
- 25 サポートファイル生成部
- 26 ソース書換部
- 27 コンパイラ
- 28 リンカ
- 29 実行形式プログラム記憶領域
- 31 ソースファイル
- 32 プリプロセッサ
- 33 ソース解析部
- 34 変数データベース
- 35 ヘッダファイル
- 36 サポートプログラムソースファイル
- 37 ソース書換
- 38 プリプロセッサ
- 39 コンパイル
- 40 リンク
- 41 実行形式プログラム

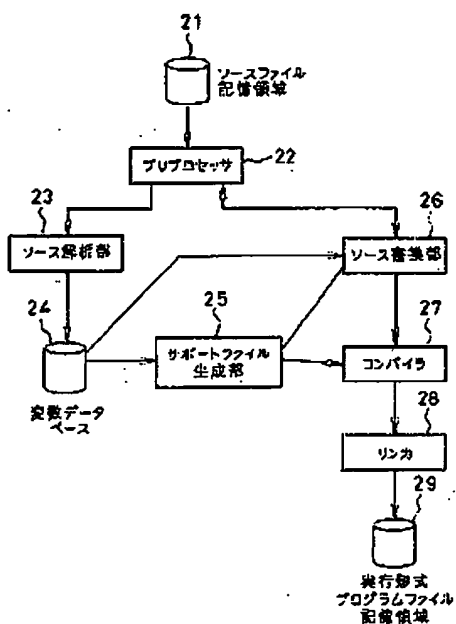
(7)

特開平11-272475

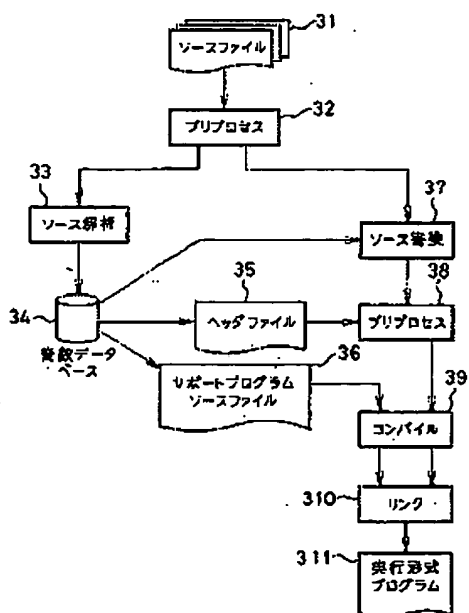
【図1】



【図2】



【図3】



【図6】

```

int a;
int b=20;
const int c=100;

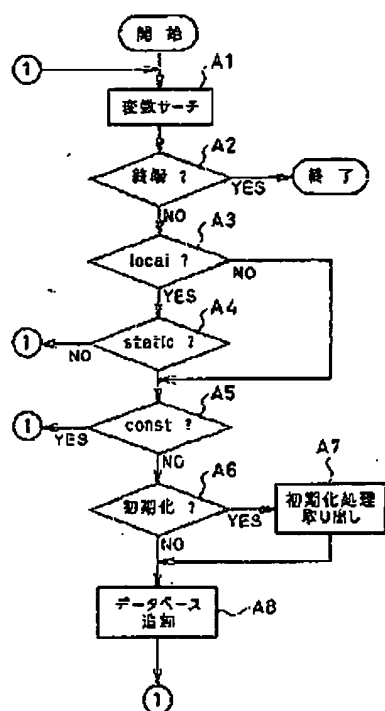
void sub(void)
{
    static char d=5;
    int e=10;

    a=30;
    d=40;
}
  
```

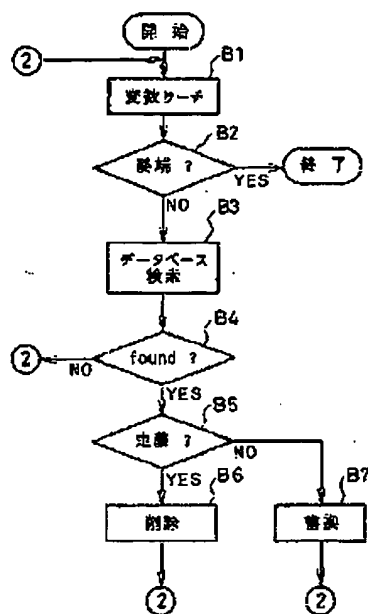
(8)

特開平11-272475

【図4】



【図5】



【図7】

型	スコープ	変数名	初期化
int	global	a	-
int	global	b	=20
char	sub()	d	=5

【図8】

```

/* gdefn */
/* グローバル変数及びスタティック変数の
代り変数構造体 */
struct GSData {
    int_global_w;
    int_global_b;
    char_sub_d;
};
extern struct GSData *gsdat(void);
#define GSData0 (gsdat(0))
  
```

(9)

特開平11-272475

【図9】

```

/*gsinit.c*/
#include "gsdef.h"

/*初期化処理*/
void Init GSData(struct StaticData *p)
{
    p->global_b=20;
    p->sub_d=6;
}

/*代替定数値取得処理(タスク毎に固有)*/
struct GSData * Get GSData(void)
{
    struct GSData *p;
    p=/"タスクのID値からタスク毎のGSDataのテーブルを探す"/
    if(p)/*もしテーブルに登録されていなかったら*/
        p=/"新規に割り当て、テーブルに登録する"/
        InitGSData(p);/*そして、初期化処理を行う*/
    }
    return p;
}

```

【図10】

```

#include"gsdef.h"

const int c=100;

void sub(void)
{
    int a=10;

    GSData(.global_a)=30;
    GSData(.sub_d)=40;
}

```

【手続補正音】

【提出日】平成10年11月16日

【手続補正1】

【補正対象書類名】図面

【補正対象項目名】図6

【補正方法】変更

【補正内容】

【図6】

(10)

特開平 1 1-272475

```
int a;  
int b=20;  
const const int c=100;  
  
void sub(void)  
{  
    static char d=5;  
    int e=10;  
  
    a=30;  
    d=40;  
}
```

*

```

/* gaderfn */
/* グローバル変数及びスタティク変数の
代管理構造体 */
struct GSDData {
    int _global_w;
    int _global_bs;
    char _sub_d;
};

extern struct GSDData *gsdat(void);
#define GSDAT(x) (gsdat()->(x))

```

【手続補正2】

【補正対象音類名】図面

【補正対象項目名】図8

【補正方法】変更

【補正内容】

【图8】

【手続補正3】

【補正対象音類名】図面

【補正対象項目名】図9

【補正方法】変更

【補正内容】

【图9】

*

```

/* *main.c/*
#include "gtdrf.h"

/*初期化処理*/
void Init GSData(struct SisdData *p)
{
    p->gsdata_b=20;
    p->gsdata_o=3;
}

/*代替実行機迄の取得処理(タスク毎に固有)*/
struct GSData * Get GSData(void)
{
    struct GSData *p;
    p=/* タスクID番号からタスク毎のGSDataのテーブルを探す */
    if(p==/* 0はテーブルに登録されていなかった */
    p=/* 表頭にあり当て、テーブルに登録する */
    InitGSDatap(p);/*として、お蔵化処理を行う */
}

return p;

```